Cross-VM Network Channel Attacks and Countermeasures within Cloud Computing Environments

Atif Saeed, Peter Garraghan and Syed Asad Hussain

Abstract—Cloud providers attempt to maintain the highest levels of isolation between Virtual Machines (VMs) and inter-user processes to keep co-located VMs and processes separate. This logical isolation creates an internal virtual network to separate VMs co-residing within a shared physical network. However, as co-residing VMs share their underlying VMM (Virtual Machine Monitor), virtual network, and hardware are susceptible to cross VM attacks. It is possible for a malicious VM to potentially access or control other VMs through network connections, shared memory, other shared resources, or by gaining the privilege level of its non-root machine. This research presents a two novel zero-day cross-VM network channel attacks. In the first attack, a malicious VM can redirect the network traffic of target VMs to a specific destination by impersonating the Virtual Network Interface Controller (VNIC). The malicious VM can extract the decrypted information from target VMs by using open source decryption tools such as Aircrack. The second contribution of this research is a privilege escalation attack in a cross VM cloud environment with Xen hypervisor. An adversary having limited privileges rights may execute Return-Oriented Programming (ROP), establish a connection with the root domain by exploiting the network channel, and acquiring the tool stack (root domain) which it is not authorized to access directly. Countermeasures against this attacks are also presented

Index Terms—Cloud Computing, Virtual Machine Monitor, Cross-VM attack, Network-Channel attack, ROP, Impersonation.

1 Introduction

Cloud computing has risen in prominence due to its service model enabling elastic on-demand access to computing resources and now underpins modern business operations. Cloud computing security is an important concern for enterprises when they shift critical information to geographically distributed cloud platforms that are directly not under their jurisdiction of control. There are many security concerns for cloud computing as it utilizes different technologies spanning networks, databases, operating systems, virtualization, resource scheduling, transaction management, load balancing, concurrency control and memory management, which are potentially vulnerable to attacks. Cloud computing security researchers work to introduce new security exploitation events or attacks that may affect the providers and users.

Cloud computing heavily leverage virtualization methodologies. Virtualization facilitates multiple operating systems (different or same) to co-reside on the same physical server concurrently. Virtualization technologies such as HyperV, Xen, KVM, and VMWare are key enablers of cloud computing systems. Their key benefit is in cost saving. Same physical hardware is offered among multiple VMs along with the ability to provide strong isolation between co-resided VMs, i.e. a guest VM cannot interfere in the operation of other guest VMs running on the same machine. Such isolation is a key support for major public providers such as Microsoft Azure, Amazon Elastic Compute Cloud (EC2), Google Compute Engine (GCE), and Rackspace [1].

However, such logical isolation is not impenetrable. A myriad of previous studies has demonstrated how co-resident VMs could be vulnerable to attacks through shared file systems [2], cache side-channels [3], [4] or through compromising hypervisor layer using rootkits [5]. Thus, the threat of cross-VM attacks remains [6], [7], [8] where an attacker uses one VM to control or access other VMs on the same hypervisor. As such, various methods have been devised for strategic VM placement in order to exploit co-residency. [9], [10].

Hypervisors virtualization attempt to realize this assumption by implementing logical isolation between VMs using traditional access-control approaches. However, it is possible for attackers to circumvent them via side-channel attacks.

Moreover the existence and threat of ROP (Return- Oriented Programming) attack in real time systems has been discussed in variety of settings. Researchers have demonstrated that if an attacker can successfully exploit the earlier version of Adobe reader and Acrobat by launching ROP, they may be able to compromise and control the victimized system [11], [12]. Furthermore, developers have developed ROP-based rootkits to compromise Window operating Systems [13]. After the execution of these rootkits, the attackers are successful in hiding a malicious process through which these rootkits manage to bypass the integrity protection system of the OS. So far, all ROP based attacks target either the applications or operating systems and do not target hypervisors directly. Hypervisors having large set of code

- *A. Saeed is with School of Computing and Communication, Lancaster University, UK.E-mail:asaeed@cuilahore.edu.pk*
- *Peter Garraghan is with School of Computing and Communication, Lancaster University, UK. E-mail:p.garraghan@lancaster.ac.uk*
- *Syed Asad Hussain is with Department of Computer Science, Comsats University Islamabad, Lahore Campus. E-mail:asadhussain@cuilahore.edu.pk*

are at high risk of bugs, due to which an attacker manages to launch an ROP attack on them.

Despite the clear potential of cross-VM attacks for exploiting shared memory and disk, exhibition of cross-VM network-channel and privilege escalation that uses the ROP in conjunction with the network-channel attack has not been demonstrated. Current network-based attacks exploit existing vulnerabilities such as ARP spoofing and DNS poisoning that are difficult to use for VM-targeted attacks [14]. The most commonly discussed network-based challenges focus on the fact that cloud providers place more layers of isolation between co-resided VMs than in non-virtualized settings because the attacker and victim are often assigned to separate segmentations of virtual networks and domains.

Cloud providers mitigate cross-VM network channel attacks by introducing the concept of isolation through an internal virtual network. Logical isolation of hardware resources can give protection against poor access-control policies, preventing VMs running on the same hardware from interfering with each other's execution or data ex-filtration. However, such logical isolation may not be appropriate if an attacker can bypass them by launching a different type of attack.

This paper proposes novel methodologies in which the attacker VM can redirect network traffic of the victim VM and set the unobtrusive destination point to receive the network traffic of the victim. It also launches a privilege escalation attack by exploiting RoP in conjunction with network-channel in cross-VM settings. The aim of these investigations is to explore whether the isolation of cloud systems, i.e., virtual machines and hypervisors, can be circumvented by the proposed attack (and if so, how?). We demonstrate that these two attacks result in successful violation of isolation properties of virtualization and escalate privilege level of non-root VMs. For responsible disclosure, all vulnerabilities found in our research have been reported to the OpenStack and Ravello security teams, we have also provided solutions for fixing the identified issues.

This paper is an extension of our conference paper [15], which proposed a novel zero-day network channel attack for redirecting the traffic of other co-located VMs. In this attack, the created dummy interface impersonates a TAP (Test Access Point) device. Impersonated TAP device in combination with the network mirror exploits the network channel. The network mirror then redirects the network traffic of other co-located VMs to the desired destination point. This approach is further extended through another novel zero-day privilege escalation attack in a cross VM cloud environment. It works by escalating the privilege level of non-root VMs. The exploitation of Return Oriented Programming (ROP) in conjunction with the network channel is used to launch this attack. Non-Root VM will hijack and control the *ToolStack* of the hypervisor, from this it can control all other co-located VMs. This paper also discusses the countermeasures for these attacks.

The rest of this paper is organized as follows: section 2 describes the background, section 3 presents related work for cross-VM attacks. Cloud system model and network traffic flow in OpenStack and Xen hypervisor are discussed in section 4. Section 5 presents the attack methodology. Section 6 presents the experimentation setup followed by the evaluation in Section 8. Countermeasures are discussed in section 8. Section 9 concludes the paper.

2 Background

Cross-VM attacks and their ability to exploit shared resources to extract or leak the sensitive information from co-located VMs have been investigated by many researchers. There has been an assumption that collocated VMs which share network interfaces trust each other [16]. This assumption is challenged due to the sharing of physical hardware in public clouds and the existence of co-residency attacks [9].

Zhang et al. [17] categorized co-residency attacks into three types of side-channel classes; *access driven* side-channel attack, *time driven* side-channel attack and *trace driven* side-channel attack. Access driven side-channel attack exploit shared micro-architectural modules such as caches. A time-driven side-channel attack occurs when the total execution times of cryptographic operations with a fixed key are influenced by the value of the key. Trace driven attack captures a profile of cache activity. In [5], [18] researchers have demonstrated methods to leak sensitive data through TCP/IP. Ranjith et al. [19] provides a method for using timing channel for data leakage.

There are numerous methods that are used to harden hypervisors i.e, Xen, which in turn enhances system security. Trustvisor [20] is a special-purpose thin hypervisor that offers code reliability as well as data privacy for an application. Another category of thin hypervisor is Bitvisor [21] that is designed to focus on the security of I/O devices. However, it has a very small piece of code that helps in reducing the jeopardy of attacks at runtime, the main drawback of its functionality is that, it supports only one VM. This is the reason that it has very limited usage. NOVA [22] uses microkernel-like access to virtualization level by moving maximum functionality to the user level. It is designed and implemented with very little performance overhead by using modern hardware virtualization. But there is still a room for improvement.

Researchers have also revealed some methods that are used to protect running commercial hypervisors i.e., Xen, KVM, etc. without compromising their functionalities [23], [24]. HyperGuard and HyperCheck [25] place the hypervisor measurement agent (MA) in the CPU System Management Mode (SMM) available in x86 system to separate MA from the hypervisor it defends. However, MA can measure the reliability of the hypervisor code, its security modules confined as SMM does not deliver all the relative data required by the MA. HyperSentry [23] resolves such complexities by placing MA in the hypervisor but uses SMM to protect it.

HyperSafe [24] is a specially designed hypervisor for protection against the control-flow hijacking attack. It uses features of the hardware to implement a non-bypassable memory lockdown technique through which only a special routine in the hypervisor can allow to write in the memory. Moreover, it also implements more fine-grained control-flow integrity using a strict pointer base indexing structure through which all function calls in the hypervisor are converted to point in a special table.

There is a number of different real-time attacks that manipulate ROP systems. On the application layer, Adobe has declared that a critical vulnerability has existed in Adobe Flash Player 10.0.45.2 and earlier versions [11]. These vulnerabilities also occur in Adobe Reader and Acrobat 9.x. These vulnerabilities are exploited by applying ROP, it bypasses data execution prevention (DEP) [26], a security system implemented by Windows, that may compromise the full system and the attacker can potentially take control of the victimized system [13].

In [27], the authors proposed an offline automated framework for auditing consistent isolation between virtual networks in the OpenStack cloud. Cloud insider attack detector and locator (CIADL) on multi-tenant network isolation for OpenStack is presented in [28]. The authors [29] presented an approach to attack on the Xen hypervisor utilizing return-oriented programming (ROP). It modifies the data in the hypervisor that controls whether a VM is privileged or not and thus can escalate the privilege of an unprivileged domain (domU) at run time.

3 Related Work

Researchers have identified a number of cross-VM attacks. In [14] the authors categorized the network channel attacks into three different categories; spoofed ARP, virtual hub, and ARP Poisoning. The attacking VM launches an ARP spoofing attack by forging an identical IP address within the target VM and sends an ARP request to the virtual router. The virtual router updates the routing table when the spoofed ARP request is received. As a result, any traffic directed to a target VM is instead mistakenly sent to the attacking VM, which can then decide to either perform sniffing or modification. In bridge network configuration mode, the bridge acts as a virtual hub. All VMs share the virtual hub to communicate with the network. An attacking VM is able to sniff the virtual network by using a sniffing tool, Wireshark [30]. In router network mode [16], the router plays the role of a virtual switch using a dedicated virtual interface to connect each VM. Here, a malicious VM can undertake ARP poisoning [31], thus redirecting the packets towards itself, and then sniffing packets going to and coming from other VMs.

ROP-based rootkits have been offered in the Windows operating system at the kernel layer. SecVisor [1] upon execution can manage to hide the malicious processes, files and network connections in Windows. These rootkits can circumvent kernel integrity protection systems.ROP technique can be used to exploit Apple iPhone [32] in which an unauthorized user installs applications or leak customers SMS database [33].

In [34], the authors presented the ZombieLoad attack which uncovers a novel Meltdown-type effect in processor's previously unexplored fill-buffer logic. Their analysis shows that faulting load instructions (i.e., loads that have to be reissued for either architectural or micro-architectural reasons) may transiently de-reference unauthorized destinations previously brought in the fill buffer by the current or a sibling logical CPU. Hence, they reported data leakage of recently loaded stale values across logical cores.

In [35], the authors proposed *MemJam*, which utilizes aliasing to establish a side-channel attack that exploits the false dependency of memory read-after-write events and provides a high-quality intra-cache line timing channel. As a proof of concept, they demonstrated the first key recovery attacks on constant-time implementations of all symmetric block ciphers supported in the current Intel integrated performance primitives (Intel IPP) cryptographic library: triple DES, AES, and SM4.

All these methodologies are based on the measurement of code reliability, which can be helpful in detecting the attacks for modifying the hypervisor code or injecting external malicious code. So, the proposed attack model that utilizes ROP cannot be detected by such defense mechanisms. This is because in the proposed attack scheme no such injection of external code or modification of the hypervisor code has been performed.

3.1 Our Contribution

This paper explores the risk of cross-VM attacks, using a concrete cloud service provider (OpenStack and Azure [36], [37]) as a case study. It proposes a method of privilege escalation by exploiting the ROP method in conjunction with a network channel in the cross-VM setting of a virtualized system.

The authors in [38] leveraged an approach in which an unprivileged VM using the ROP technique can manage to modify the code of a hypervisor through which it can escalate its privilege level. However, there are still a number of potential channels to explore. The authors have not suggested any indication for the privilege escalation of bonding of ROP and exploitation of network channels. This work focuses on the exploitation of network channels for a variety of reasons: (i) it arguably has the highest potential to escalate the privilege level of unprivileged VM, (ii) prior work only escalates the privilege level using traditional approaches that can now be easily countered in virtualization. The proposed approach provides an in-depth analysis of the techniques used and their effects, both qualitative and quantitative in various settings, (iii) the result of this proposed approach is so effective that if it is successful it can not only terminate other VMs running on the same physical hardware but can also launch a DOS attack.

To the best of our knowledge, no work yet has demonstrated redirection of a co-residing target VM's network traffic, by exploiting the network channel through a combination of mirroring and impersonation along with the exploitation of cross-VM network channel through ROP. This is used in conjunction with a side-channel attack. This approach is a first step towards using ROP to attack the leading cloud providers in cross-VM settings. Moreover, as discussed next, the features of mounting the proposed attack in a virtualized system are only successful if the proposed attack settings are successful.

4 System Model and Network Traffic Flow in OpenStack cloud and Hypervisor (Xen)

Investigation of our approach has been conducted in OpenStack deployed within Xen hypervisor. Before presenting the details, we define the fundamental concepts of OpenStack and Xen.

4.1 OpenStack

OpenStack is an open-source platform for cloud deployment. It attempts to provide sufficient compatibility to Amazon's external interfaces. It provisions scalability as well as a wide diversity of hypervisors, network infrastructures, and storage systems. The OpenStack section that is used in this research is neutron [39] that enables the network component and is available under the Apache-2 license.

OpenStack offers a cloud service with an infrastructure-as-a-service (IaaS) model with many forms of paired services. Each service offers an application programming interface (API) that facilitates this integration. It offers the options to developers to rent virtual machine instances in a pay-as-you-go manner from its data center. The types of instances are micro, small, large, and extra-large in size, and each type has different capabilities for memory, I/O, and CPU. Among all the instances types, a small instance is allocated to a single virtual CPU (vCPU). In addition, multiple small instances share a single physical machine. Thus, small instances are used in our experiments to evaluate the proposed attack.

OpenStack can easily be connected with other 3rd party tools like Open vSwitch (OVS), an open-source virtual switch, for smooth functioning. The main purpose of implementing the OVS in OpenStack is to support advance network features such as layer-2 switch and offers different features of Access Control List, VLAN, and subnet or private network to VMs.

4.2 OpenStack Architectural Description

Following are the node specifications and their descriptions used in OpenStack architecture.

Controller: Responsible for executing the services of management software that are needed for functioning of OpenStack platform.

Compute: Compute nodes execute virtual machine instances in Open-Stack. KVM is used as a hypervisor in this node. This node is also responsible for providing firewall services. One can deploy more than one compute node in a setup.

Network: The responsibilities of network nodes ensure the creation of virtual networks needed by the customers to create public or private networks. It connects their virtual machines with the external networks, i.e. the Internet.

Deployment: All these nodes can be configured in the form of a single machine or multiple machines as shown in Figure 1. In a single node setup, all nodes are deployed within the same physical machine for facilitating all VMs and networking capability. In a multi-node setup, a single controller is responsible for system management of all compute nodes executing VMs.

4.3 Xen

Xen is a hypervisor that provides strong isolation between VMs co-residing on the same physical machine. It is open-source (GPLv2) and is operated by Xen.org, a cross-industry organization [40]. The Xen architecture is shown in Figure 2.

Xen can be configured with any leading cloud provider. In this study, we have configured Xen underneath OpenStack. A brief description of OpenStack Xen architecture and its domains is as follows:

Privileged and unprivileged domains: A number of VMs or domains can run on Xen. When the system boots, bootloader first loads Xen which loads its main domain called dom0. As Xen does not manage any device drivers, dom0 is used to access the hardware and can handle device requests. It is also the only administrative domain responsible for creating, pausing, unpausing, saving, and destroying other VMs residing on the same physical hardware. Contrary to this, other domains that dom0 launches have no such admin privileges and are called unprivileged domain (domU). The data that controls whether a domain is privileged or not is a boolean value in a domain structure within the hypervisor code(set as 1 for dom0 and 0 for domU). If we want to escalate the privilege of a domU, we need to modify this allocated value for this domU from zero to one. The OpenStack architecture having Xen running underneath is shown in Figure 3.

Toolstack and Console: Domain 0 consists of a special management tool called control stack (also termed Toolstack) that lets a user to manage virtual machine creation, destruction, and configuration.

4.4 Cloud Network Architecture

Each VM has an assigned IP address visible to virtual switch and all VMs are connected to *br-int*, i.e. open virtual switch (OVS), *br-int* and *br-ex* are OpenVSwitches which are responsible for managing ingress and egress requests respectively. OVS is a virtualized switch which is a part of the hypervisor.

A virtual switch is similar to a virtual network interface that is organized by combining one or more virtual Ethernet interfaces. A virtual switch is further connected to a virtual router which is used for maintaining a routing table for ingress and egress traffic. Routers are implemented with the strict configuration of the firewall (FWaaS) that implements a security perimeter. The router further transmits the packets to its next hop, i.e. on the br-ex. The br-ex contains a physical network interface, eth0 which finally passes the packets on the external network, i.e. the Internet.

There are four types of distinct virtual networking devices TAP devices, veth pairs, Linux bridges (L.B), and Open vSwitch bridges as shown in Figure 4. The Ethernet frame is transmitted from the vnic of a VM to the Internet, go through the devices; TAP, bridges, veth pair, Open vSwitch bridge within the host.

Hypervisors such as KVM and Xen configure a virtual network interface card (VIF or vNIC) using a TAP device. An Ethernet frame transmitted towards the TAP device is received by the guest operating system.

A **veth pair** is a pair that is directly connected to virtual network interfaces i.e., vNIC. This pair acts as a network cable. Veth pairs are used as virtual patch cables to create connections between virtual bridges.

A **Linux bridge (L.B)** acts like a switch that follows the MAC learning principle. Multiple network interfaces devices can be connected to such bridges. There are MAC caching tables

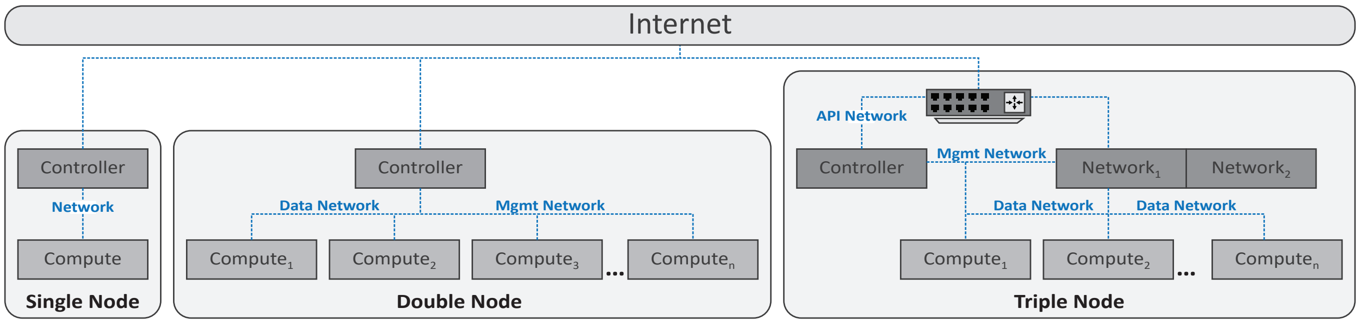


Fig. 1. OpenStack Setups.

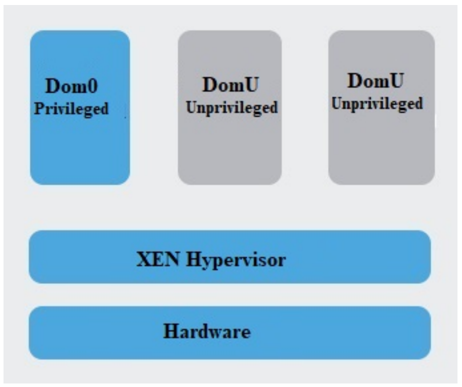


Fig. 2. Xen Hypervisor Architecture [40]

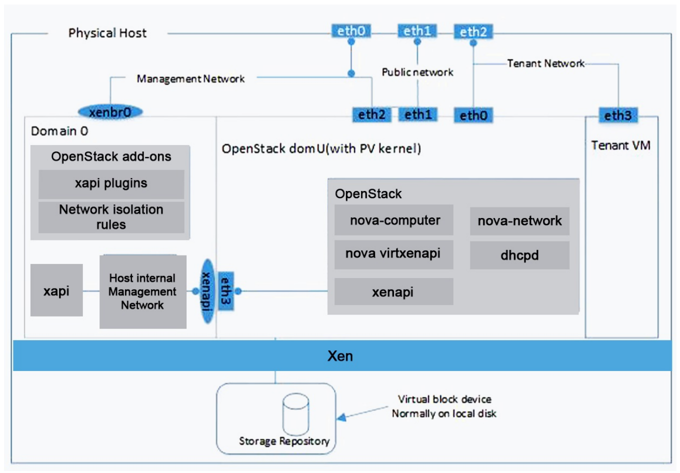


Fig. 3. OpenStack Xen Architecture [41]

in these bridges that are used to keep records of which interfaces on the bridge are communicating with a host on the link. An Ethernet frame arrives at an interface connected to the bridge, the host MAC address and port on which the frame received is saved in a MAC caching table for a limited time. When the bridge needs to send a frame, it first looks up the table to see if the destination address of the frame is saved or not. If so, the Linux bridge in such a case will forward the frame only through port where the frame is received. If its record is not saved in a table, then the bridge will flood the frame to all network ports, with the exclusion of the port where the frame received.

An **Open vSwitch** bridge works like a virtual switch. The network interface devices are connected to the Open vSwitch bridge's ports, and the configuration of the ports are similar to the physical switch port, including VLAN configuration.

The **Integration bridge** i.e. br-int is an Open vSwitch bridge. All guest VMs running on the host connects to this bridge. Cloud providers use advance networking techniques to implement isolation between the guest VMs by configuring the br-int ports. The integration bridge is responsible for doing VLAN tagging and un-tagging of network traffic coming to and from VMs.

4.5 Network Flow

The VM creates data and keeps it on the VNIC (Virtual NIC card) that is associated with VM's eth0. The data is then transmitted to the TAP (Test Access Point) device on the compute host. Generally, a TAP offers a path to access the data passing through a network. The TAP devices are further linked to the Linux bridges that pass the data to the veth pair which acts as a one-side of the cable. Data sent to one side of the veth pair can be received at the other end. The other end of the pair is on the integration bridge (br-int). This bridge is responsible for attachment of all the VM TAP devices and any other bridge on the system.

OVS int-br further connects with br-eth1 to pass the traffic to an external interface. The int-br-eth1 from br-int is responsible for connecting with phy-br-eth1 of br-eth1. The int-br-eth1 is one-half part of a veth pair connecting with phy-br-eth1 bridge that is the other half of veth pair, which manages VLAN systems is trunked over the physical ethernet device i.e., eth1. VLAN configured at each interface keeps network traffic of all VMs separated from each other. VLAN ID has been assigned at br-int and traffic of each VLAN follows its own path.

5 The Attacks

The main focus of this research is to exploit the network channel of cloud computing by launching two zero-day attacks. Attack methodologies, their challenges, limitations, and stages are discussed in this section.

5.1 Impersonation Attack

There is a potential to observe network traffic of the target machine on a network-channel. The main objective of

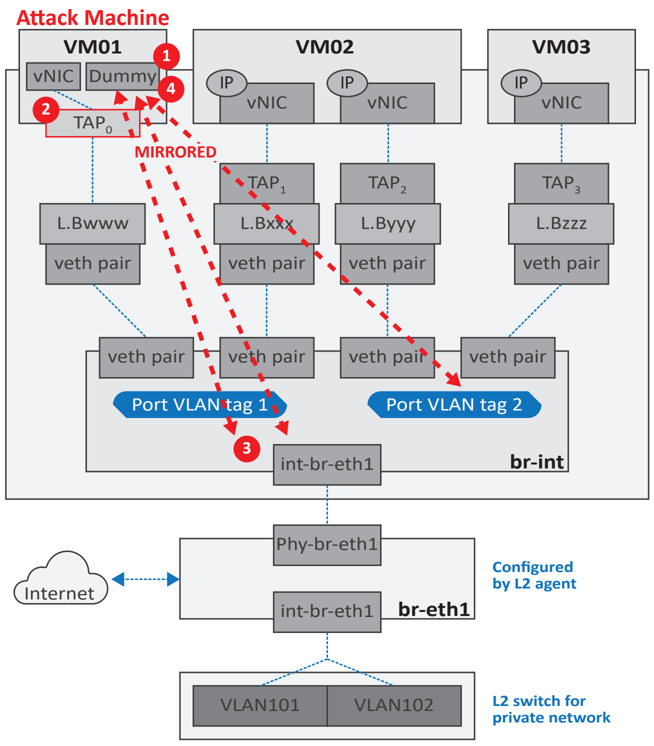
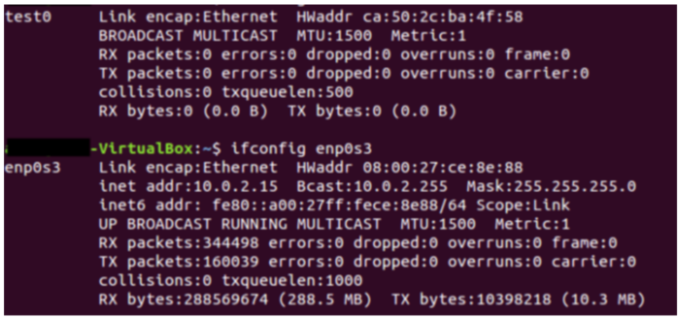


Fig. 4. Impersonation Attack

this attack is to compromise the internal interface of the network, i.e. OpenVswitch (OVS) through which all VMs are connected. By compromising the OVS, an attacker is capable of penetrating within the running system. To do so, the attacker needs to investigate the internal interface of the network, i.e. the bridge interface through which the traffic of all VMs passes.**(1)** Firstly, a dummy network device is configured in an attacking VM that does not have an active NIC adapter installed or disconnected from the network. Secondly, it is impersonated into the TAP device. By removing all types of interface identity, the network card behaves like a TAP. The main reason for this impersonation is that TAP devices cannot be used to attach network namespaces to the bridge. To overcome this gap, Ethernet is needed to be impersonated with TAP. **(2)** Connectivity requests are sent to the bridge that presumes it to be a valid TAP interface and is automatically added as a conventional TAP device. This bridge also contains the interfaces of other VMs. Moreover, the identity of this dummy interface is hidden by removing its footprints, so that it cannot be traced on the network routes. **(3)** A network mirror is set up in an unobtrusive way at this interface and the dummy interface is configured as a destination point. The mirror redirects the network traffic passing through it to or from the port onto the set destination point. **(4)** This attack hides all the details about routing and the host information. This is achieved by assigning 0 to this dummy interface. Zeros remove all the identities, i.e. IP and route information assigned to this interface. The attack scenario and its steps are shown in Figure 4. Cloud providers permit bridging of a TAP interface that has no valid private Ethernet interface at the backend is the main vulnerability in the network architecture [15].


```
test0     Link encap:Ethernet  HWaddr ca:50:2c:ba:4f:58
          BROADCAST MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:500
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

[redacted]-VirtualBox:~$ ifconfig enp0s3
enp0s3    Link encap:Ethernet  HWaddr 08:00:27:ce:8e:88
          inet addr:10.0.2.15  Bcast:10.0.2.255  Mask:255.255.255.0
          inet6 addr: fe80::a00:27ff:fece:8e88/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:344498 errors:0 dropped:0 overruns:0 frame:0
          TX packets:160039 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:288569674 (288.5 MB)  TX bytes:10398218 (10.3 MB)
```

Fig. 5. TAP vs NIC

A cloud provider does not allow the bridging of a TAP interface that does not have a private Ethernet interface at the backend. But in the 'TAP Impersonation attack', the attacker misleads the cloud provider by going through the aforementioned steps. The difference between the TAP and regular devices can be seen in Figure 5. In this Figure, test0 is a TAP device and enp0s3 is a regular network device.

5.2 Privilege Escalation Attack

There is a potential to take over the control of Toolstack of hypervisor at dom0 through the attacking machine which resides at domU. This experiment circumvents the security perimeter of OpenStack by introducing the concept of Return oriented Programming (ROP) technique in conjunction with the exploitation of network channel.
The basic idea of Return-oriented programming is as follows:
Return-oriented programming is an effective code-reuse methodology in which an attacker executes code that is already present in the address space of a cloud model to undermine the security perimeter [42]. This attack works by escalating the privilege level of non-root VMs. The exploitation of Return Oriented Programming (ROP) in conjunction with the network channel is used to launch this attack. In this attack, code-reuse methodology has been applied in which malicious VM copies the code of OpenStack in its own memory/domain. By doing so, it re-uses the code of the actual open stack in its own memory. The reason for this activity is that OpenStack allows the guest VM to further sublet their resources. To manage their own resources, guest VMs need to be a host of their own machine. Due to this reason, OpenStack allows the guest VMs to copy the code.

Figure 6 illustrates the general ROP attack based on code-reuse instructions. It shows a simplified version of a program's memory layout consisting of a code section, libraries (lib), a data section, and a control structure (CS) section. To mount an ROP attack based on code-reuse, the attacker exploits a buffer overflow vulnerability of a specific program. Hence, the attacker is able to overflow the local buffer and overwrite adjacent control-flow information of the CS section (step 1). In Figure 6, the attacker injects a subset of OpenStack code (code-reuse) whereas program execution is redirected to code i.e., to an instruction sequence in the lib section (step 2). The instruction sequence of the linked library is executed until a subset of OpenStack code has been reached which redirects the execution to the

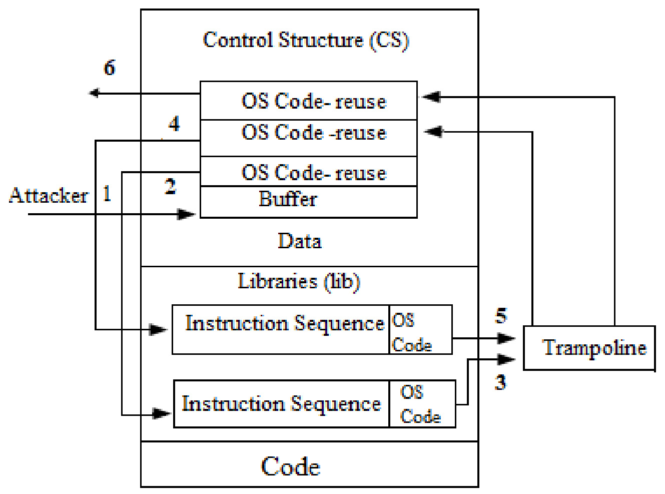


Fig. 6. Return Oriented Programming

next sequence of instructions by using a trampoline (step 3). The trampoline is also part of the linked libraries and is responsible for loading the address of the next instruction sequence from the CS section and redirecting execution to it (step 4). This procedure is repeated until the attacker terminates the program.

It has been shown that by compromising the network channel, the attacker manages to penetrate into the domain i.e. dom0 of a running system where the privilege VM resides.

In OpenStack cloud having Xen architecture running underneath, the root or admin VMs reside at dom0 and are able to access the underlying hardware directly. These root/admin VMs have special privileges i.e., they are able to control other VMs. Dom0 is the first machine activated by the system. There is a special stack (called Toolstack) in this domain that allows a VM to manage virtual machine creation, destruction, and resource allocation. Guest VMs residing in domU (an unprivileged domain) cannot directly access the hardware. These guest VMs cannot interfere with other guests' operations and management. In guest VM, which is an attacking machine, a special feature of OpenStack named as OpenStack client has been configured. The main concept behind the configuration of this special feature of OpenStack is that, through this feature,**(1)** a malicious VM copies the code of OpenStack in its own domain i.e. domU. By doing so, the technique of code-reuse is applied in its internal memory. This in-memory code is already testified and scrutinized within security perimeter of OpenStack and is already in an active state and hence security perimeters are unable to block it. The main benefit of code-reuse scheme in a guest VM is that this malicious guest VM becomes a host of its own local machine (sub-host from main root). After becoming a host of its own local machine, it is granted a "dom0" for managing its own sub-guest VMs. This model looks like a tree in which there is a main root that resides in dom0 and have further guest VMs belonging to domU. Among these guest VMs, one guest VM acts as a root of its local system which has its own guests VMs, but globally it acts as a sub-root. This malicious guest VM avail both the

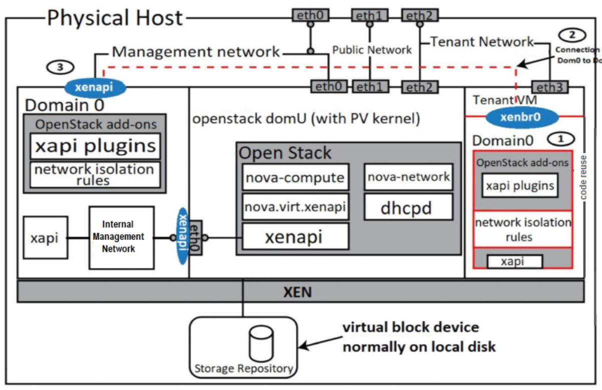


Fig. 7. Privilege Escalation Attack

privileges (dom0 and domU). Dom0 manage its own local guest VMs which can be further sublet and domU is added by its inherent characteristic. There is a special python API in dom0 i.e., xapi that make a connection between other domains through network bridge i.e. br0. To make a connection between domU and dom0, a special python API i.e., XenAPI at domU is used to connect with xapi of dom0. This connection follows a network path from ethernet device to bridge and it uses internal management network to reach from domU to dom0. Figure 7 shows concept of attack model for OpenStack Xen architecture.

In this attack scenario, **(2)** an adversary establishes a connection with dom0 by connecting a veth bridge pair that connects xapi with xapi of python library. In veth bridge pair, one bridge already exists in main root and other bridge is part of the malicious guest VM that the attacker acquires by applying code reusing technique. Whenever xapi connects with other xapi through bridge, it presumes that root of one cloud attempts to connect with root of other cloud for cloud expansion or hybrid cloud and it automatically allows the connection. **(3)**Due to this, attacking machine penetrates into dom0, gain root privileges and takes over control of tool stack to manage guest VMs. After getting control of Toolstack the attacker is not only able to manage other guest VMs running on same physical machine but also can cause DoS attack by creating unnecessary VMs and assigning them huge resources.

6 Experimental Setup

Previous section presents the attacks settings and methodologies through which (i) a malicious VM can utilize network channel to learn about the traffic of a co-resident VM and (ii) a non-privilege VM can escalate its privilege level by applying ROP in conjunction with a network channel. It has been shown that after exploitation of an internal network bridge, it allows an attacker to determine when other VMs are transmitting traffic. Leakage of such information appears secure, but it can be pretty valuable to attackers. The attack is launched through penetration into the current system, surreptitious redirection of the network traffic and even the combination of mirroring and impersonation attack. These steps have been practically checked on OpenStack cloud testbed. The other experiment shows that after the privilege escalation, an attacker takes over the control of Toolstack through which it can manage other co-resided VMs. Controlling Toolstack is not only valuable to attackers but also a serious threat to other VMs and cloud providers. The steps that have been introduced to execute this attack are, penetration into the root domain, illegal possession of Toolstack and the usage of ROP in combination with network channel OpenStack cloud testbed.

6.1 Experimental Settings

These experiments have been evaluated on OpenStack local testbed that is configured with different types of setups. Three guest VMs are launched and all VMs are assigned IP addresses of different classes. Two types of networks that have been configured in the experiments are NAT and bridge network. In impersonation attack, NAT network has been configured. VMs have been assigned local IP addresses of 10.1.1.6, 10.1.1.7 and the attacking machine is assigned an IP address of 10.1.1.8 respectively. However in ROP attack, bridge network has been configured through which VMs assigned real-time IPs. In our experiment, VM and the host machine have been assigned IP addresses of 192.168.10.8 and 192.168.10.1 respectively. Each guest holds two VCPUs, co-residing on a quad-core processor, specifically an Intel Core 2 Q9650 with an operating frequency of 3.0 GHz. All guest VMs are separated by using internal network to ensure isolation. One guest VM acts as the attacking machine and the others as the target. KVM is configured as the hypervisor. Different flavors of operating systems i.e. Ubuntu 16.04 server with a Linux kernel 4.8 and Cirros are configured. Cirros is the tiny branch of Ubuntu. The size of memory in all guest VMs are large (1-3GB) enough to meet requirements of the experiment. In general, the attacker can either passively wait for the victim VM to start communicating and then spy messages, or actively spy on communication by redirecting its traffic. In our experiments, we consider the scenario which is in attacker's advantage, in this situation co-located VMs (VM2 and VM3) communicate with each other by sending ping command.

It is also pertinent to mention that the experimental settings are a realistic secure setup for virtualized environments. In fact, in a cloud datacenter isolation among multiple VMs may improve the security as one VM cannot interfere with the operations of other VMs.

6.2 Network setup

OpenStack offers the facility of public and private types of networks. IP addresses assigned to VMs from the public network are accessed externally which is through the Internet. The private network is only used for internal cloud communication and cannot be accessed from the Internet.

6.3 Security Configuration

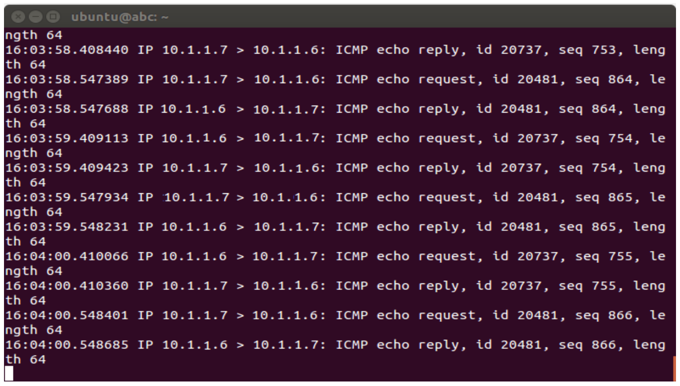
Attackers can intrude in a system at any point, hence multiple levels of security are required in a network to secure data. In such an environment, cloud providers cannot fully trust the traditional physical network security alone. They need to add their own security rules to secure the cloud. Security models hence need to be investigated and change from centralized to distributed and from physical to virtual configuration. Networking uses IPtables to achieve security group functions. It enables the IPset option that uses a hash table to improve security group's performance. When a new port is created, an additional IPset option is added to its iptables chain. If a member of a security group is changed, iptables rules are reloaded. However, by enabling the IPset option, iptables are not needed to be reloaded, if only the members of the security group are changed, it should just update IPset rules.

A security group has been presented to avoid change of settings when VM's security group is created. These settings are initiated by the individuality check of a new security group name. A table of a security group that implements such a group has a VM ID field as a primary key and a security group ID. This is the identifier of a security group.

These two zero-day attacks are based on the exploitation of network channel and their hypothesis areas:

Hypothesis 1: The penetration of external devices in current running virtualized system.

Hypothesis 2: Attacking machine belonging to *domU* should penetrate into *dom0* and control the *tool stack*. Through this, it can exploit other VMs running on the system.



```
ngth 64
16:03:58.408440 IP 10.1.1.7 > 10.1.1.6: ICMP echo reply, id 20737, seq 753, leng
th 64
16:03:58.547389 IP 10.1.1.7 > 10.1.1.6: ICMP echo request, id 20481, seq 864, le
ngth 64
16:03:58.547688 IP 10.1.1.6 > 10.1.1.7: ICMP echo reply, id 20481, seq 864, leng
th 64
16:03:59.409113 IP 10.1.1.6 > 10.1.1.7: ICMP echo request, id 20737, seq 754, le
ngth 64
16:03:59.409423 IP 10.1.1.7 > 10.1.1.6: ICMP echo reply, id 20737, seq 754, leng
th 64
16:03:59.547934 IP 10.1.1.7 > 10.1.1.6: ICMP echo request, id 20481, seq 865, le
ngth 64
16:03:59.548231 IP 10.1.1.6 > 10.1.1.7: ICMP echo reply, id 20481, seq 865, leng
th 64
16:04:00.410066 IP 10.1.1.6 > 10.1.1.7: ICMP echo request, id 20737, seq 755, le
ngth 64
16:04:00.410360 IP 10.1.1.7 > 10.1.1.6: ICMP echo reply, id 20737, seq 755, leng
th 64
16:04:00.548401 IP 10.1.1.7 > 10.1.1.6: ICMP echo request, id 20481, seq 866, le
ngth 64
16:04:00.548685 IP 10.1.1.6 > 10.1.1.7: ICMP echo reply, id 20481, seq 866, leng
th 64
```

Fig. 8. Traffic capturing of Attacking VM

7 Evaluation

This section evaluates working of both attack experiments discussed in Section 5.1 and 5.2.

7.1 Impersonation attack

By applying an attack strategy discussed in Section 5.1, attacking VM1 can observe the traffic between target VMs as shown in Figure 8. The attacking VM receives an ICMP echo reply. The echo request and echo reply show successful communication between two of the VMs.

The attack is effective when it is possible to determine the sender and receiver communication between target VMs (e.g., packet header source IP address).

Figure 9 shows VM network traffic prior, during, and after the attack. Experiment time between 0 - 30 minutes depicts that attacking VM1 exhibits random network traffic not different from that of VM2 and VM3 (point A). The attack commences at 25 minutes (point B), where it is observable that attacking VM1 receives substantial network

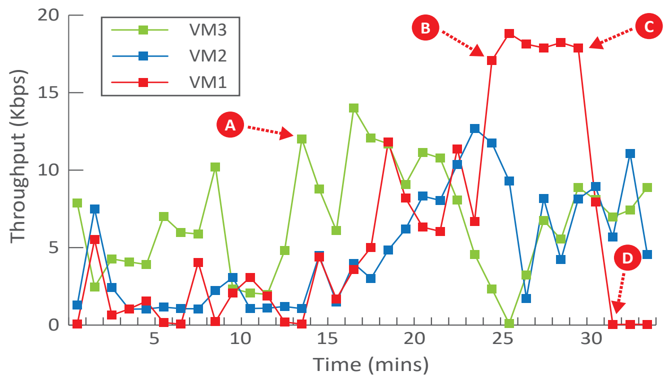


Fig. 9. Network Traffic of Co-residing VM

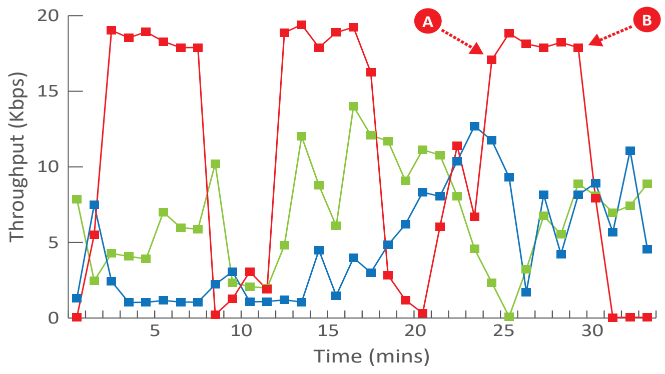


Fig. 10. Cyclic Behavior of attacking VM

traffic compared to target VMs, and continues to do so for 6 minutes until attack completion (point C). The reason for this sudden increase in throughput is because all target VM network traffic is being redirected through the attacking VM.

While comparing network traffic of VM1 with other VMs during the entire experiment, it is possible that the cloud administrator could detect this as anomalous behavior due to the sudden spike in network usage. This may lead to the deployment of a countermeasure to restrict VM network traffic that reaches a defined threshold. However, we believe that in the context of cloud computing such a countermeasure would be challenging to implement. Firstly in many public cloud settings, VM resource usage is seen as a black box by the provider. If resource demands do not violate resource capacity requested by a customer, this is seen as typical of acceptable behavior. Secondly, even if the system is attempting to monitor typical resource patterns using bandwidth monitoring tools such as prtg [43], countermeasures will likely include a time delay for determining irregular resource patterns. Therefore, if a VM is compromised even for a few minutes this may give sufficient time to an attacker to achieve his/her objectives. For example, in 2008 the defense solution of a system operated by the Georgia Government during an HTTP attack [44] remained activated for 5 minutes after the attack was launched. Finally, the detection of typical network traffic patterns becomes incredibly difficult if the attacking VM is capable of creating cyclical network patterns before an attack, as shown in Figure 10. The potential detection strategy for such attacks is to place a Network Intrusion Detection System (NIDS) [45] at points within the network to monitor traffic to and from all sources on the network. It performs traffic analysis at different time intervals and compares the traffic to find out the attacks. Once an abnormal behavior in network traffic is observed, an alert can be sent to the administrator. But in this case, the attacking machine is generating the same amount of traffic periodically and in the third peak executes an attack following a similar resource pattern seen previously (point A to B). Hence, the abnormality in the traffic pattern cannot be observed. In this particular case, ideally, the administrator would need to scan all inbound and outbound traffic, but the limitation in doing so might create a bottleneck that would affect the overall performance of the network.

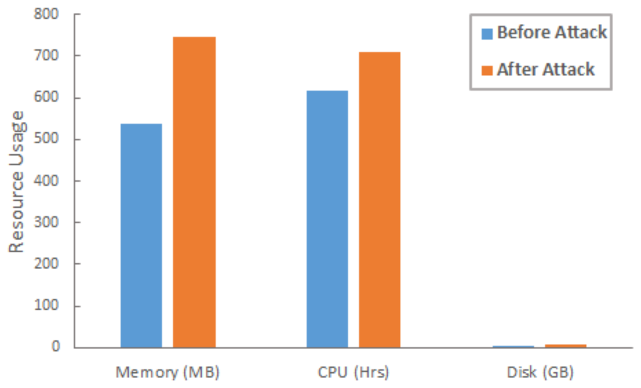


Fig. 11. Resource Comparison before and after attack

7.2 Privilege Escalation

Table 1 shows the total resource allocation and resource utilization statistics of each VM for one-week. The value of resource utilization mentioned in Table 1 is not fixed; it varies from time to time depending upon the VM and the programs running on them.

To demonstrate the effectiveness of the proposed approach, the attacking scheme described in section 5.2 is applied in an open-source cloud platform i.e. OpenStack, and Microsoft Azure. For each cloud platform, the experimental setting described in section 6.1 has been implemented. Three VMs have been launched to perform this experiment. Initially, we have assigned some basic resources to all these three VMs as shown in Table 1. After the execution of a privilege escalation attack, a run-time measurement has been performed to observe the resource utilization of attacking VMs. Figure 11 compares resource utilization of the attacking VM before and after the implementation of the attack. Blue lines show resource utilization before the attack. The main reason behind the variation in resource utilization is that attacking VM uses ROP to escalate its privilege level, hence it puts more processing load by executing the same set of code in its local memory.

A network analyzer tool, prtg [43] is used to analyze the network traffic of attacking VM. The prtg is a third-party tool and can be configured on any open-source platform. To evaluate the experiment setting, prtg has been configured in the OpenStack host machine to monitor the network traffic

TABLE 1
Resource Allocation and Utilization of Each VM.

Co-located VMs	Memory (MB)	Disk (GB)	CPU (Hrs)	Memory (MB)	Disk (GB)	CPU (Hrs)
VM1	1024	15	1	745.8482	6.25	709.45
VM2	2048	25	2	937.6591	9.13	1054.764
VM3	3072	40	3	3109.675	17.45	989.243

of co-residing VMs.

Figure 12 shows the output of a network traffic packet analyzer of an attacking VM. The IP configurations of attacking VM and the host machine are 192.168.10.8 and 192.168.10.1 respectively.

Figure 12 exhibits that the attacking VM can access the host machine which it is not authorized to do so through a network channel. There is no logic for an attacking VM to access the host machine as they have been assigned their own gateways and routing paths for ingress and egress network traffic.

Figure 13 shows the I/O graph of an attacking VM. I/O graph is a convenient tool for measuring the network traffic and throughput of any selective VM. This graph demonstrates that high network traffic is transmitted from the attacking VM to the host machine and very low traffic from the host machine is received. The routing table of host and attacking VM has separate routes and there are no intersection communication points between these two machines, hence there is no logic for attacking VM to communicate with the host machine with such a high network traffic. Orange line in this graph shows the network traffic received by the attacking machine while blue line shows the network traffic sent out to the host machine. The line graph indicates that the attacking VM is attempting to communicate with the host machine with a very high traffic.

8 Countermeasures

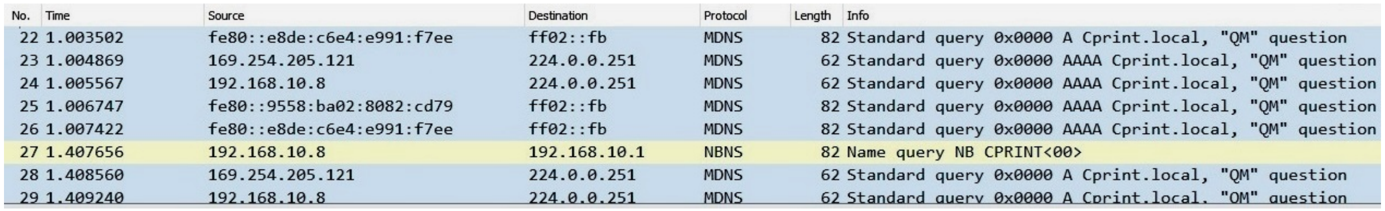
8.1 Impersonation Attack

The main defense against this attack might seek to modify the open-source code of OpenStack cloud to at least limit the granularity of network-based side-channels. The main step that helps in launching the experimented attack is penetration in the network. We can inhibit this attack effectively if we are successful in restricting the penetration of any external device in the current running system. Multiple VMs' interfaces are ultimately connected to the OpenStack internal network bridge, i.e. br-int that further connects with a physical device for connection with the Internet. As described already, the attacker manages to place itself in the internal network through an impersonated TAP interface and these TAP interfaces do not have any private Ethernet interfaces. The security check ensures to only bridge TAP interfaces that have some private Ethernet interfaces which are protected through a security perimeter or IP table rules. The vulnerabilities and assumptions that make such compromise possible are because of, cloud providers that allow bridging of a TAP interface with the same Ethernet which is used to connect to the Internet. This creates a serious security problem. The direct connection of the TAP interface that does not have any private Ethernet at the backend with the bridge provides an opportunity for an attacker to penetrate into the network.
The modification in the open-source code of OpenStack ensures elimination of data leaks by restricting direct connection of all the TAP Interfaces with a bridge that accesses the Internet with the same Ethernet. The analysis of the interface has revealed that each valid interface has three attributes: - tag, interface, and type. The tag indicates that VLAN is enabled to ensure the isolation between VMs, interface exhibits its association with backend private Ethernet and type shows the behavior of the interfaces. The security checks need to ensure all these three attributes of TAP play their role before connecting with the bridge. This strategy can restrict penetration of the attacker into the running system, thus providing an effective defense against network-channel attacks.

8.2 Privilege Escalation Attack

The strategy to limit the privilege escalation attack modifies open-source code of the OpenStack cloud. The main phase that helps in launching the privilege escalation attack is the reuse of the code to circumvent the security perimeter which ultimately penetrates into the root domain of the system. This attack can be inhibited if we are successful to thoroughly scrutinize the code and restrict penetration into the running system.

A new security perimeter has been introduced that defends the system in a way akin to a network firewall. The job of such a security perimeter is to protect/block unauthorized users from accessing the root domain. The underlying logic of the newly introduced security perimeter examines the internal state of the code and identifies the malicious connectivity of unauthorized users. Already existing security rules cannot prevent such attacks because the codes for these rules are already stored in the internal memory of the system, and the security perimeter already examined such codes, so it treats all of these codes the same as the original ones and thus has already checked.
To expand the cloud network, establishment of a root-to-root connection is required. Such connections are established through Xen special API called *xapi*. The proposed security API upon connection request will internally check *xapi* and whether is there any dual registration of this VM or not. A connection request will be declined if it has dual registration otherwise it grants the connection. A special API has been implemented in the proposed security perimeter that will check on run-time whether this *xapi* has any *Xenapi* connection in its local domain? If it finds any *Xenapi* connection, its connection request would be blocked otherwise connection will be granted. The only limitation of applying this security perimeter is network delay. Figure 14a shows the normal root-to-root connection for cloud expansion. Figure 14b shows enabling of security check for root-to-root connection through xapi. This security check will examine



```
No. Time      Source                     Destination   Protocol  Length Info
 22 1.003502  fe80::e8de:c6e4:e991:f7ee  ff02::fb      MDNS      82 Standard query 0x0000 A Cprint.local, "QM" question
 23 1.004869  169.254.205.121            224.0.0.251   MDNS      62 Standard query 0x0000 AAAA Cprint.local, "QM" question
 24 1.005567  192.168.10.8               224.0.0.251   MDNS      62 Standard query 0x0000 AAAA Cprint.local, "QM" question
 25 1.006747  fe80::9558:ba02:8082:cd79  ff02::fb      MDNS      82 Standard query 0x0000 AAAA Cprint.local, "QM" question
 26 1.007422  fe80::e8de:c6e4:e991:f7ee  ff02::fb      MDNS      82 Standard query 0x0000 AAAA Cprint.local, "QM" question
 27 1.407656  192.168.10.8               192.168.10.1  NBNS      82 Name query NB CPRINT<00>
 28 1.408560  169.254.205.121            224.0.0.251   MDNS      62 Standard query 0x0000 A Cprint.local, "QM" question
 29 1.409240  192.168.10.8               224.0.0.251   MDNS      62 Standard query 0x0000 A Cprint.local, "QM" question
```

Fig. 12. Traffic Capturing through Host Machine

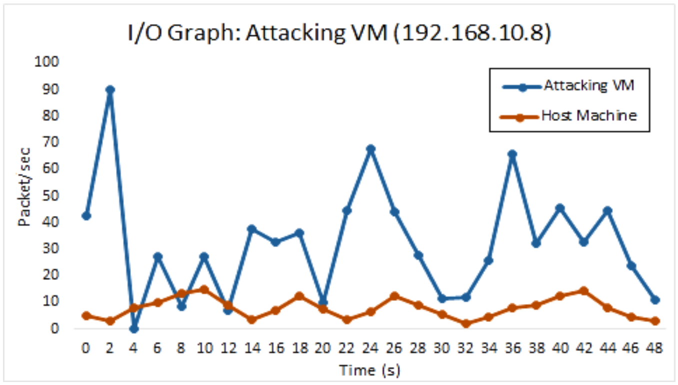


Fig. 13. I/O Graph between an attacking VM and Host Machine

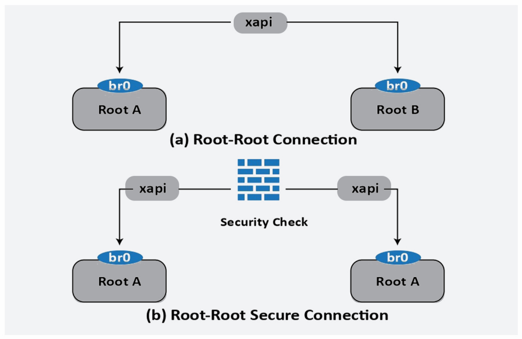


Fig. 14. (a): Root-Root Connection (b) Root-Root Secure Connection

the internal code of xapi and then decides to allow or block the connection.

9 Conclusion

This research demonstrates two successful zero-day cross-VM network attacks within a leading cloud platform i.e OpenStack. The first attack applies a combination of impersonating a TAP interface and a network mirror in the bridge interface. The consequence of this combination allows attackers to successfully redirect and monitor target VM network traffic within the same physical machine unbeknownst to customers. The second attack exploits ROP in conjunction with a network channel for the escalation of the privilege level of non-root VMs. This exploitation allows a non-root VM to make a connection with the root VM and control Tool Stack from where it can manage other co-located VMs.

The countermeasure solutions of these two zero-day attacks have also been presented. They block penetration of any external device into the system and properly scrutinize the connection request before granting the root connection.

We have highlighted the challenge for cloud providers to observe and detect such attacks due to an attacking VM which neither violates assigned VM resource capacity nor it makes any illegal connection with the root user for privilege escalation. Future work will focus on improving the current heuristics that prevent penetration of external devices into the network and also observe the request for root-connection. Furthermore, we will investigate how to overcome the challenges of distinguishing between normal resource patterns and target attacks.

References

[1] A. Seshadri *et al.*, "Secvisor: A tiny hypervisor to provide lifetime kernel code integrity for commodity oses," in *ACM SIGOPS Operating Systems Review*, vol. 41, pp. 335–350, ACM, 2007.
[2] http://cve.mitre.org/cgibin/cvename.cgi?name=CVE-2008-0923.
[3] T. Ristenpart, E. Tromer, H. Shacham, and S. Savage, "Hey, you, get off of my cloud: exploring information leakage in third-party compute clouds," in *Proceedings of the 16th ACM conference on Computer and communications security*, pp. 199–212, ACM, 2009.
[4] Z. Wu, Z. Xu, and H. Wang, "Whispers in the hyper-space: High-speed covert channel attacks in the cloud.," in *USENIX Security symposium*, pp. 159–173, 2012.
[5] J. Rutkowska, "Subverting vistatm kernel for fun and profit," *Black Hat Briefings*, 2006.
[6] D. Hyde, "A survey on the security of virtual machines," *Dept. of Comp. Science, Washington Univ. in St. Louis, Tech. Rep*, 2009.
[7] S. R. Kumari and V. Kathiresan, "Virtual environment security-considerations & practices," *Networking and Communication Engineering*, vol. 3, no. 2, pp. 87–92, 2011.
[8] S. Zhang, "Deep-diving into an easily-overlooked threat: Inter-vm attacks," tech. rep., Technical Report). Manhattan, Kansas: Kansas State University, 2012.
[9] A. Bates, Mood, *et al.*, "On detecting co-resident cloud instances using network flow watermarking techniques," *International Journal of Information Security*, vol. 13, no. 2, pp. 171–189, 2014.
[10] V. Varadarajan *et al.*, "A placement vulnerability study in multi-tenant public clouds,"
[11] Adobe, "Adobe systems. security advisory for flash player, adobe reader and acrobat: Cve-2010-1297.." http://www.adobe.com/support/security/advisories/apsa10-01.html, 2010.
[12] S. Ragan. http://www.thetechherald.com/articles/Adobe-confirms-Zero-Day-ROP-used-to-bypass-Windows-defenses/11273/.
[13] R. Hund, T. Holz, and F. C. Freiling, "Return-oriented rootkits: Bypassing kernel code integrity protection mechanisms," in *USENIX Security Symposium*, pp. 383–398, 2009.
[14] C. Modi, D. Patel, B. Borisaniya, A. Patel, and M. Rajarajan, "A survey on security issues and solutions at different layers of cloud computing," *The journal of supercomputing*, vol. 63, no. 2, pp. 561–592, 2013.
[15] A. Saeed *et al.*, "A cross-virtual machine network channel attack via mirroring and tap impersonation," in *IEEEInternational Conference on Cloud Computing (CLOUD)*, pp. 606–613, IEEE, 2018.

[16] H. Wu, Y. Ding, C. Winer, and L. Yao, "Network security for virtual machine in cloud computing," in *Computer Sciences and Convergence Information Technology (ICCIT), 2010 5th International Conference on*, pp. 18–21, IEEE, 2010.
[17] Q. Zhang, L. Cheng, and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of internet services and applications*, vol. 1, no. 1, pp. 7–18, 2010.
[18] S. J. Murdoch and S. Lewis, "Embedding covert channels into tcp/ip," in *International Workshop on Information Hiding*, pp. 247–261, Springer, 2005.
[19] P. Ranjith, C. Priya, and K. Shalini, "On covert channels between virtual machines," *Journal in Computer Virology*, vol. 8, no. 3, pp. 85–97, 2012.
[20] N. Q. Z. Z. A. D. V. G. Jonathan M. McCune, Yanlin Li and A. Perrig., "Trustvisor: Efficient tcb reduction and attestation. in ieee symposium on security and privacy, 2010.."
[21] T. Shinagawa *et al.*, "Bitvisor: a thin hypervisor for enforcing i/o device security," in *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pp. 121–130, ACM, 2009.
[22] U. Steinberg and B. Kauer, "Nova: a microhypervisor-based secure virtualization architecture," in *Proceedings of the 5th European conference on Computer systems*, pp. 209–222, ACM, 2010.
[23] A. M. Azab, P. Ning, Z. Wang, X. Jiang, X. Zhang, and N. C. Skalsky, "Hypersentry: enabling stealthy in-context measurement of hypervisor integrity," in *Proceedings of the 17th ACM conference on Computer and communications security*, pp. 38–49, ACM, 2010.
[24] Z. Wang and X. Jiang, "Hypersafe: A lightweight approach to provide lifetime hypervisor control-flow integrity," in *Security and Privacy (SP), 2010 IEEE Symposium on*, pp. 380–395, IEEE, 2010.
[25] J. Wang, A. Stavrou, and A. Ghosh, "Hypercheck: A hardware-assisted integrity monitor," in *International Workshop on Recent Advances in Intrusion Detection*, pp. 158–177, Springer, 2010.
[26] DEP, "Microsoft. data execution prevention (dep). http://support.microsoft.com/kb/875352/en-us/, 2006."
[27] J. Zhan, X. Fan, J. Han, Y. Gao, X. Xia, and Q. Zhang, "Ciadl: cloud insider attack detector and locator on multi-tenant network isolation: an openstack case study," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–23, 2019.
[28] T. Madi, Y. Jarraya, A. Alimohammadifar, S. Majumdar, Y. Wang, M. Pourzandi, L. Wang, and M. Debbabi, "Isotop: auditing virtual networks isolation across cloud layers in openstack," *ACM Transactions on Privacy and Security (TOPS)*, vol. 22, no. 1, pp. 1–35, 2018.
[29] B. Ding, Y. Wu, Y. He, S. Tian, B. Guan, and G. Wu, "Return-oriented programming attack on the xen hypervisor," in *2012 Seventh International Conference on Availability, Reliability and Security*, pp. 479–484, 2012.
[30] https://www.wireshark.org/.
[31] H.-C. Li *et al.*, "Analysis on cloud-based security vulnerability assessment," in *e-Business Engineering (ICEBE), 2010 IEEE 7th International Conference on*, pp. 490–494, IEEE, 2010.
[32] V. Iozzo, "Ralf-philipp weinmann & vincenzo iozzo own the iphone at pwn2own," 2010.
[33] B. Ding *et al.*, "Return-oriented programming attack on the xen hypervisor," in *Availability, Reliability and Security (ARES), 2012 Seventh International Conference on*, pp. 479–484, IEEE, 2012.
[34] M. Schwarz *et al.*, "Zombieload: Cross-privilege-boundary data sampling," *arXiv preprint arXiv:1905.05726*, 2019.
[35] A. Moghimi *et al.*, "Memjam: A false dependency attack against constant-time crypto implementations," *International Journal of Parallel Programming*, vol. 47, no. 4, pp. 538–570, 2019.
[36] https://www.openstack.org/.
[37] https://azure.microsoft.com/.
[38] B. Ding, F. Yao, Y. Wu, and Y. He, "Improving flask implementation using hardware assisted in-vm isolation," in *IFIP International Information Security Conference*, pp. 115–125, Springer, 2012.
[39] OpenStack, "OpenStack Networking ("Neutron")." [Accessed: Jan-18].
[40] https://www.xenproject.org.
[41] https://wiki.openstack.org/wiki/XenServer/XenAndXenServer.
[42] L. Davi and othes, "Privilege escalation attacks on android," in *international conference on Information security*, pp. 346–360, Springer, 2010.
[43] https://www.paessler.com/prtg.
[44] A. Kozlowski, "Comparative analysis of cyberattacks on estonia, georgia and kyrgyzstan," *European Scientific Journal, ESJ*, vol. 10, no. 7, 2014.
[45] B. R. Raghunath *et al.*, "Network intrusion detection system (nids)," in *International Conference on Emerging Trends in Engineering and Technology. ICETET'08.*, pp. 1272–1277, IEEE, 2008.

Atif Saeed is an active computer scientist and has published over 10 peer-reviewed articles in the field of networks and distributed systems. His primary research expertise is studying the complexity of Cyber Security, Hacking ,Cloud computing, Data-centers, Internet of Things, Networks. He is an active member of Communication and Networks research at COMSATS University Islamabad, Lahore campus. Atif Saeed received his doctorate degree from Lancaster University UK.

Peter Garraghan is a Lecturer (Assistant Professor) in Distributed Systems at Lancaster University. He is the leader of the Evolving Distributed Systems Laboratory (EDS Lab). Peter has industrial experience building production distributed systems at scale. His research interests include Cloud computing, Machine Learning Systems, Energy-efficiency, Dependability, and Resource Management.

Syed Asad Hussain received the master's degree from Cardiff University, UK and the Ph.D. degree from Queen's University Belfast, UK. He was the Head of the Computer Science Department, COMSATS University Islamabad Lahore, Pakistan, from August 2008 to August 2017. He has been serving as the Dean of Faculty of Information Sciences and Technology since 2015. He is currently leading communications and networks research at COMSATS University Islamabad Lahore. He is supervising Ph.D. students at COMSATS University and split-site Ph.D. students at Lancaster University, UK in the fields of cloud computing and cybersecurity. He was funded for his Ph.D. by Nortel Networks UK at Queen's University Belfast. He has taught at Queen's University Belfast, Lahore University of Management Sciences (LUMS), and the University of the Punjab. He was awarded prestigious endeavour research fellowship for his post doctorate at The University of Sydney, Australia, in 2010, where he conducted research on VANETs. He regularly reviews IEEE, IET, and ACM journal articles.